Proceedings of the International Multiconference on Computer Science and Information Technology pp. 131–136
ISBN 978-83-60810-14-9
ISSN 1896-7094

Applying Emphasized Soft Targets for Gaussian Mixture Model Based Classification

Soufiane El Jelali, Abdelouahid Lyhyaoui and Aníbal R. Figueiras-Vidal*
Dept. of Signal Processing and Communications
Univ. Carlos III de Madrid
Av. de la Universidad 30
Leganés, Madrid 28911, Spain
Email: {soufiane, abdel, arfv}@tsc.uc3m.es

***Abstract*—When training machines classifiers, it is possible to replace hard classification targets by their emphasized soft versions so as to reduce the negative effects of using cost functions as approximations to misclassification rates. This emphasis has the same effect as sample editing methods which have proved to be effective for improving classifiers performance. In this paper, we explore the effectiveness of using emphasized soft targets with generative models, such as Gaussian Mixture Models, that offer some advantages with respect to decision (prediction) oriented architectures, such as an easy interpretation and possibilities of dealing with missing values. Simulation results support the usefulness of the proposed approach to get better performance and show a low sensitivity to design parameters selection.**

I. Introduction

TRAINING machine classifiers with conventional search procedures using hard targets results in minimizing a cost function which is not more than an approximation to the error rate. Fisher type formulations [1] and "decision based" algorithms [2] are also approximations as well as "energy functions" [3], while the Perceptron Rule [4] has convergence difficulties and generalization limitations.

Sample selection or sample editing methods try to compensate the suboptimal character of the conventional approaches by paying more attention to samples that are more important for defining the classification borders. [5] was the first proposal along this line. It was followed by many schemes that were proposed to emphasize sample populations that pay attention to samples lying (approximately) near the borders [6][7][8] or to samples which offer higher errors [9][10]. Some original formulations about boosting schemes to construct classifier ensembles [11][12][13] seem to be rather based on giving importance to erroneous samples, although [14][15] prove that Real Adaboost emphasizes both erroneous samples and those that are near the border; the last reference also propose some generalizations. It is also clear that emphasizing sample populations is very similar to selecting appropiate cost functions. In this sense, the Maximum Margin algorithm used to train Support Vector Machines [16][17][18] can be considered as a procedure which emphasizes both kinds of samples, as well. It is unclear which of these types of samples is more important to get a good design, although the answer seems to be problem dependent [19].

Among other alternatives, the one of substituting the "hard" classification targets for soft versions is interesting, because it leads to an estimation (or regression) problem, in which standard cost functions can be applied without the difficulties mentioned above. This will allow the possibility of directly applying machines that are conceived in order to solve regression problems, such as Gaussian Processes [20][21], for classification tasks. Although there are many forms of generating soft versions of decision targets, such as using convolutional smoothing [22], it seems reasonable to keep the advantages that sample emphasis provides. This was the orientation of [23] and of our previous work [24], in which we demonstrated the effectiveness of using an emphasized combination of the original targets and those provided by an auxiliary classifier to get better classification performance for the most popular family of discriminative (predictive) designs, Multi-Layer Perceptrons (MLP).

In this paper, we extend our studies to generative models, presenting results for the well known Gaussian Mixture Models (GMM), that we apply both in discriminative and soft-target based generative forms to solve classification problems. This study is important not only for the exploration of the general applicability of soft targets approaches, but also because GMM are relatively easy to understand and they accept well principled methods to deal with missing values.

The rest of the paper is organized as follows. In Section 2, we review the basic algorithm that defines the emphasized soft targets. In Section 3, we explain how to solve a classification problem using the soft target definition and GMM models. Section 4 is devoted to check the performance of the proposed approach in a series of experimental settings. We close the paper with the conclusion which emerges from the experiments and some suggestions for further research along this line.

II. A Method to Construct Emphasized Soft Targets

In order to create soft targets, it is not practical to use the output of a previously trained (auxiliary) classifier, because the

*This work has been partially supported by MEC Pjt. TEC2005-00992. The work of S. El Jelali was also supported by an AECI grant.

decision results of the scheme being trained with these soft targets can be worse than those of the auxiliary machine, because we are trying to repeat its errors. A convex combination of the original (hard) targets and auxiliary machine outputs $\mathrm{o}_{aux}(\mathbf{x})$ can serve to avoid this difficulty; and even more, if we use a well selected local combination parameter $\lambda(\mathbf{x})$ [24], we will have the possibility of applying an emphasis to the samples. A reasonably simple procedure consists on defining soft targets

$$t_s(\mathbf{x}) = \lambda(e(\mathbf{x}))t(\mathbf{x}) + (1 - \lambda(e(\mathbf{x})))\mathrm{o}_{aux}(\mathbf{x}). \tag{1}$$

where $\mathbf{x}$ is the (training) sample, $t(\mathbf{x})$ is the original (hard) target, $\mathrm{o}_{aux}(\mathbf{x})$ is the output of the auxiliary classifier (in this paper, we will use an MLP for this role), $e(\mathbf{x})$ is the error corresponding to this auxiliary output, and $\lambda(e(\mathbf{x}))$ is a convex combination (emphasis) weight having the form

$$\lambda(e(\mathbf{x})) = \begin{cases} \exp(-\dfrac{(|e(\mathbf{x})| - \mu)^2}{\alpha_1}) \text{for} & |e(\mathbf{x})| \le \mu, \\ \exp(-\dfrac{(|e(\mathbf{x})| - \mu)^2}{\alpha_2}) \text{for} & \mu < |e(\mathbf{x})| \le 2. \end{cases} \tag{2}$$

μ, α_1, α_2 being parameters of the Gaussian bells, that must be selected during the design phase.

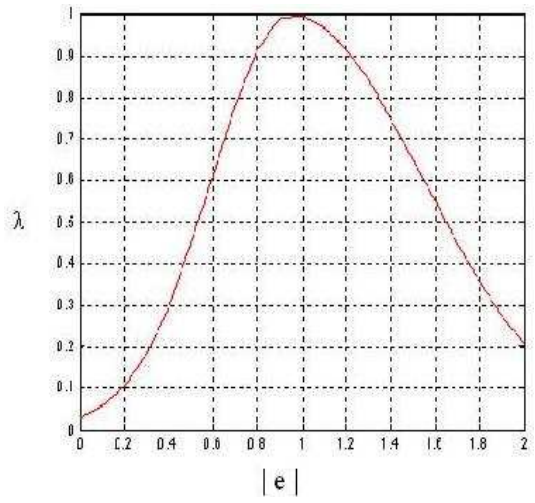

Fig. 1. Form of $\lambda(|e(\mathbf{x})|) : \mu = 1,\ \alpha_1 < \alpha_2$

The form we propose for the convex combination weight is shown in Fig.1. Note that μ is the value of $|e(\mathbf{x})|$ at which $t_s(\mathbf{x})$ is maximum (unity), and that α_1 and α_2 control the decay of $t_s(\mathbf{x})$ from this value when samples are clearly well classified ($|e(\mathbf{x})| \to 0$), so they are not very important, or when the samples give highly erroneous results ($|e(\mathbf{x})| \to 2$), in which case they are difficult to classify correctly. So, we can select the value of $|e(\mathbf{x})|$ at which we apply the highest emphasis, and we have flexibility to reduce this emphasis for both "easy" or "impossible" samples.

Although it is obvious that there are many other flexible and reasonable forms for (2), it is easy to verify, as predicted in [25], that the performance of the corresponding designs does not depend critically on the particular emphasis which is employed.

III. Using GMM

To solve a classification problem by means of GMM models directly, we have to construct these models for both hypotheses, $\widehat{p}(\mathbf{x}|i) = \sum_l \pi_l p_l(\mathbf{x}|i), i = \pm 1$ ($\{p_l(\mathbf{x}|i)\}$ being Gaussian densities), typically using the Expectation-Maximization (EM) algorithm; then, we construct the approximation to the Maximum A Posteriori (MAP) classifier

$$\widehat{P}r(1|\mathbf{x}) \underset{-1}{\overset{+1}{\gtrless}} \widehat{P}r(-1|\mathbf{x})$$

where

$$\widehat{P}r(i|\mathbf{x}) = \frac{\widehat{p}(\mathbf{x}|i)\widehat{p}(i)}{\sum_{i'} \widehat{p}(\mathbf{x}|i')\widehat{p}(i')} \text{for} i = 1, -1. \tag{3}$$

$i, i' = 1, -1$; $\{\widehat{p}(i)\}$ can be obtained as relative frequencies.

On the other hand, assuming that we are working with soft target $t_s(\mathbf{x})$, the estimation is addressed under a multidimensional GMM model, first estimating the joint distribution

$$\widehat{p}(t_s, \mathbf{x}) = \sum_l \pi_l p_l(t_s, \mathbf{x}). \tag{4}$$

with the help of the EM algorithm. After it, since the a posteriori expectation of the target is an estimate of $Pr(1|\mathbf{x}) - Pr(-1|\mathbf{x})$, we look for

$$\begin{aligned} \widehat{t}_s(\mathbf{x}) &= \widehat{E}\{t_s|\mathbf{x}\} = \int t_s \widehat{p}(t_s|\mathbf{x}) dt_s \\ &= \int t_s \frac{\widehat{p}(t_s, \mathbf{x})}{\widehat{p}(\mathbf{x})} dt_s = \sum_l \frac{\pi_l p_l(\mathbf{x})}{\sum_{l'} \pi_{l'} p_{l'}(\mathbf{x})} \int t_s \widehat{p}_l(t_s|\mathbf{x}) dt_s \\ &= \sum_l \frac{\pi_l p_l(\mathbf{x})}{\sum_{l'} \pi_{l'} p_{l'}(\mathbf{x})} \widehat{E}_l\{t_s|\mathbf{x}\}. \end{aligned} \tag{5}$$

where marginal densities are

$$p_l(\mathbf{x}) = \int p_l(t_s, \mathbf{x}) dt_s \text{for} l = 1, \ldots, L. \tag{6}$$

It is immediate to check that

$$\widehat{E}_l\{t_s|\mathbf{x}\} = \mathbf{w}_{l,e}^T \mathbf{x}_e = \mathbf{w}_l^T \mathbf{x} + w_{0,l}. \tag{7}$$

where $\mathbf{w}_l$ is given by the normal equations

$$\mathbf{w}_l = V_{\mathbf{x}\mathbf{x},l}^{-1} \mathbf{v}_{t_s\mathbf{x},l}. \tag{8}$$

$V_{\mathbf{x}\mathbf{x},l}$ and $\mathbf{v}_{t_s\mathbf{x},l}$ being the autocovariance matrix of data $\mathbf{x}$ and the cross-covariance vector of target t_s and data $\mathbf{x}$ under (Gaussian) model $p_l(t_s, \mathbf{x})$, that are obtained from the autocovariance matrix of this distribution by deleting its first row and column and as its first column, respectively; finally,

$$w_{0,l} = E_l\{t_s\} - \mathbf{w}_l^T E_l\{\mathbf{x}\}. \tag{9}$$

where $E_l\{t_s\}$ and $E_l\{\mathbf{x}\}$ are the first element and the remaining vector of the mean vector of $p_l(t_s, \mathbf{x})$, respectively.

Note that $V_{\mathbf{xx},l}$ and $E_l\{\mathbf{x}\}$ are the covariance matrix and mean vector of Gaussian density $p_l(\mathbf{x})$.

The above expressions are formally similar to those corresponding to Mixture of Experts (MoE) ensembles[26][27], as previously mentioned in [28]. However, in our case, we will obtain $\widehat{t}_s(\mathbf{x})$ from the parameters corresponding to the GMM of $p(t_s, \mathbf{x})$ calculated by applying the EM algorithm. Finally, the decision for a new sample $\mathbf{x}_n$ is

$$dec(\mathbf{x}_n) = sgn(\widehat{t}_s(\mathbf{x}_n)). \tag{10}$$

Fig.2 depicts the corresponding decision machine.

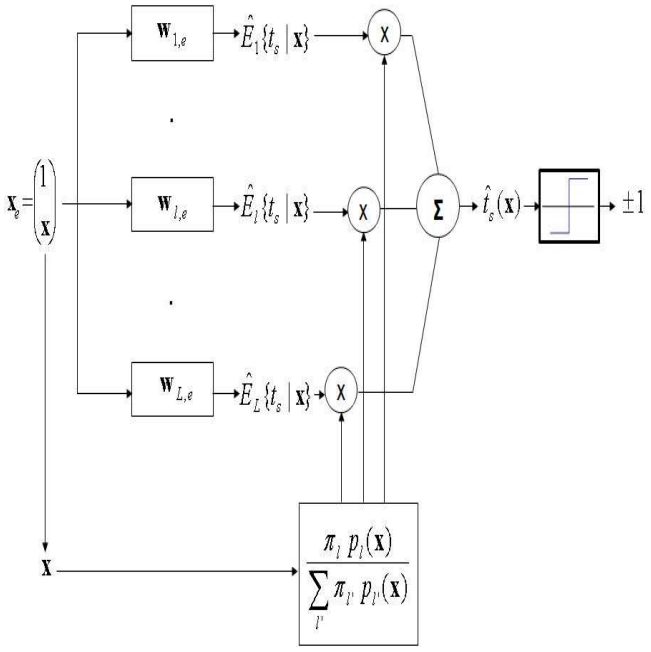

Fig. 2. Classification procedure using soft target and GMM models

IV. EXPERIMENTS

A. Datasets

We have tested the proposed method with six datasets that are frequently used as benchmarks for classification problems. The first is the synthetic bidimensional Kwok problem [29]; the other five are taken from the UCI Machine Learning Repository [30]: Abalone (converted into a binary problem according to [31]), Breast Cancer, Contraceptive, Ionosfera, and Pima Indian. We will refer to them as kwo, aba, bre, con, ion, and pim, respectively. Table 1 shows their main characteristics.

B. Training

All data were normalized -if not previously done- between -1 and 1.

We use an MLP with N hidden neurons as the auxiliary machine. The number of hidden neurons, as well as parameters μ, α_1 and α_2, and the number of components of GMM, L for the joint model and L_1, L_{-1} for the separate class models, are found by means of a 10-fold cross-validation (CV) using

TABLE I
CHARACTERISTICS OF THE TEST PROBLEMS

Dataset	Train ($\mathbf{C_{+1}}/\mathbf{C_{-1}}$)	Test ($\mathbf{C_{+1}}/\mathbf{C_{-1}}$)	#dim
kw	500 (200/300)	10200 (4080/6120)	2
aba	2507 (1238/1269)	1670 (843/827)	8
bre	420 (145/275)	279 (96/183)	9
con	883 (506/377)	590 (338/252)	9
ion	201 (101/100)	150 (124/26)	34
pim	461 (161/300)	307 (107/200)	8

10 runs, exploring the following values:

-N: 4, 6, 8, 10 ,12, 14, 16

-μ: 0.01, 0.1, 0.3, 0.6, 1, 1.2, 1.6, 2

-α_1, α_2: 0.001, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 2, 3, 4, 5

-L: 4, 5, 6, 7, 8, 9, 10

-L_1, L_{-1}: 2, 3, 4, 5

These experiments did not present important difficulties due to eventual closeness to singularity of the covariance matrices [32]; when this appears in other cases, the solution proposed in [33] can be applied.

Table 2 presents the results of the experiments, as well as the values of the design parameters that are obtained from applying CV. With respect to direct MLP and MAP GMM designs, the results of our approach (SOFT GMM) show an advantage going from very small for kwo and bre to important for pim, ion, and con. The results are also competitive with those provided by an SVM with a Gaussian kernel whose parameters have also been optimized by a 10-fold CV process (penalty factor C : $[10^{-1}11010^2 10^3 10^4]$), taking the kernel dispersion as $\sigma = \sqrt{\#dim/2}$. We have used the IRWLS SVM toolbox for Matlab [37] for pattern recognition, with tolerance $\epsilon = 10^{-5}$ to provide the accuracy required for the solution of the Quadratic Programming (QP) algorithm.

To make evident the advantage of the proposed method, Fig. 3 shows the classification boundaries for kwo obtained with the optimal MLP, the MAP-GMM method, and the proposed approach, compared with the theoretical boundary. Note that the greater proximity of the proposed classifier boundary to the optimal boundary means that it correctly classifies some samples that MLP and MAP-GMM place in the wrong decision region.

TABLE II
AVERAGED PERCENTAGES OF CORRECT CLASSIFICATION ± STANDARD DEVIATION AND DESIGN PARAMETERS FOR EACH METHOD (MLP: N; MAP GMM: k_1, k_2; SOFT GMM: $k, N, \mu, \alpha_1, \alpha_2$; AND SVM: C) FOR THE TEST DATASETS WITH A 10-FOLD CROSS-VALIDATION

Dataset	MLP	MAP GMM	SOFT GMM	SVM
kwo	83.49 ±3.32 $N = 16$	84.77 ±0.59 $k_1 = 3$, $k_2 = 3$	85.00 ±0.83 $k = 9$, $N = 6$, $\mu = 0.3$, $\alpha_1 = 0.01$ $\alpha_2 = 0.1$	83.68 ±0.59 $C = 10^3$, $\sigma = 1$
aba	78.12 ±0.54 $N = 12$	72.90 ±0.84 $k_1 = 4$, $k_2 = 4$,	73.01 ±0.67 $k = 9$, $N = 12$, $\mu = 1.2$, $\alpha_1 = 0.1$ $\alpha_2 = 0.1$	77.47 ±4.44 $C = 10$, $\sigma = 2$
bre	97.51 ±0.60 $N = 8$	94.90 ±1.46 $k_1 = 3$, $k_2 = 4$	95.34 ±0.91 $k = 8$, $N = 10$, $\mu = 0.3$, $\alpha_1 = 0.01$, $\alpha_2 = 0.05$	97.49 ±0.34 $C = 1$, $\sigma = 2.12$
con	70.38 ±2.24 $N = 10$	62.88 ±1.41 $k_1 = 4$, $k_2 = 4$	65.69 ±1.98 $k = 8$, $N = 6$, $\mu = 1.6$ $\alpha_1 = 0.1$, $\alpha_2 = 4$	70.39 ±0.65 $C = 10$, $\sigma = 2.12$
ion	93.22 ±1.48 $N = 6$	93.15 ±2.56 $k_1 = 3$, $k_2 = 3$	94.52 ±1.82 $k = 9$, $N = 6$, $\mu = 1.2$, $\alpha_1 = 0.05$ $\alpha_2 = 0.5$	97.80 ±0.45 $C = 10$, $\sigma = 4.12$
pim	78.33 ±1.33 $N = 6$	72.72 ±1.40 $k_1 = 2$, $k_2 = 2$	77.37 ±1.50 $k = 6$, $N = 4$, $\mu = 0.3$, $\alpha_1 = 0.01$, $\alpha_2 = 1$	75.44 ±0.79 $C = 10^2$, $\sigma = 2.12$

Of course, this advantage is obtained at a high training cost, since we need to select the best of $\#N \times \#\mu \times \#\alpha_1 \times \#\alpha_2 \times \#L$ designs, and not just $\#N$ (for a single MLP), $\#L_1 \times \#L_{-1}$ (for the MAP GMM case), or $\#C$ (for SVM schemes). However, once the proposed machine is trained, its application requires a pretty moderate computational effort, equivalent to that of using a GMM for regression.

An additional problem could appear in terms of sensitivity with respect to the values of the design parameters selected by CV. To explore this in a exhaustive form (considering variations of all the design parameters around each optimal design) is too complex. A more reasonable option is to present the results of the so-called "omniscient" designs, those that select the design parameters according to their performances for the test sets. Of course, these omniscient versions are not acceptable as valid designs, but comparing their performances (and the corresponding design parameter values) with those of the optimal CV designs is an indication of the sensitivity of these machines.

TABLE III
"OMNISCIENT" RESULTS: AVERAGED PERCENTAGES OF CORRECT CLASSIFICATION (± STANDARD DEVIATION) OF MLP, MAP GMM, SOFT GMM, AND SVM FOR THE TEST DATASETS. DESIGN PARAMETERS ARE ALSO INDICATED

Dataset	MLP	MAP GMM	SOFT GMM	SVM
kwo	84.30 ±0.66 $N = 6$	85.30 ±0.62 $k_1 = 2$, $k_2 = 5$	85.31 ±1.06 $k = 6$, $N = 12$, $\mu = 0.3$, $\alpha_1 = 0.1$ $\alpha_2 = 0.01$	83.68 ±0.59 $C = 10^3$, $\sigma = 1$
aba	78.20 ±0.41 $N = 6$	72.97 ±0.58 $k_1 = 5$, $k_2 = 5$	73.01 ±0.67 $k = 9$, $N = 12$, $\mu = 1.2$, $\alpha_1 = 0.1$, $\alpha_2 = 0.1$	77.42 ±4.44 $C = 10$, $\sigma = 2$
bre	97.53 ±0.56 $N = 4$	95.94 ±1.08 $k_1 = 5$, $k_2 = 4$	95.77 ±1.10 $k = 6$, $N = 10$, $\mu = 0.3$, $\alpha_1 = 0.001$, $\alpha_2 = 1.5$	97.63 ±0.39 $C = 10$, $\sigma = 2.12$
con	70.51 ±2.24 $N = 6$	63.15 ±0.95 $k_1 = 2$, $k_2 = 2$	65.69 ±1.98 $k = 8$, $N = 6$, $\mu = 1.6$, $\alpha_1 = 0.1$, $\alpha_2 = 4$	71.07 ±0.65 $C = 10^2$, $\sigma = 2.12$
ion	93.22 ±1.48 $N = 6$	93.15 ±2.56 $k_1 = 3$, $k_2 = 3$	94.54 ±1.89 $k = 10$, $N = 8$, $\mu = 1$, $\alpha_1 = 5$, $\alpha_2 = 2$	97.80 ±0.45 $C = 10$, $\sigma = 4.12$
pim	78.80 ±0.97 $N = 12$	72.89 ±1.80 $k_1 = 2$, $k_2 = 3$	78.92 ±0.98 $k = 6$, $N = 12$, $\mu = 0.3$, $\alpha_1 = 0.1$, $\alpha_2 = 0.01$	78.93 ±0.56 $C = 1$, $\sigma = 2.12$

Table 3 presents the same values than Table 2, but for the corresponding omniscient machines. Note that, even having to select much more design parameters, the proposed method

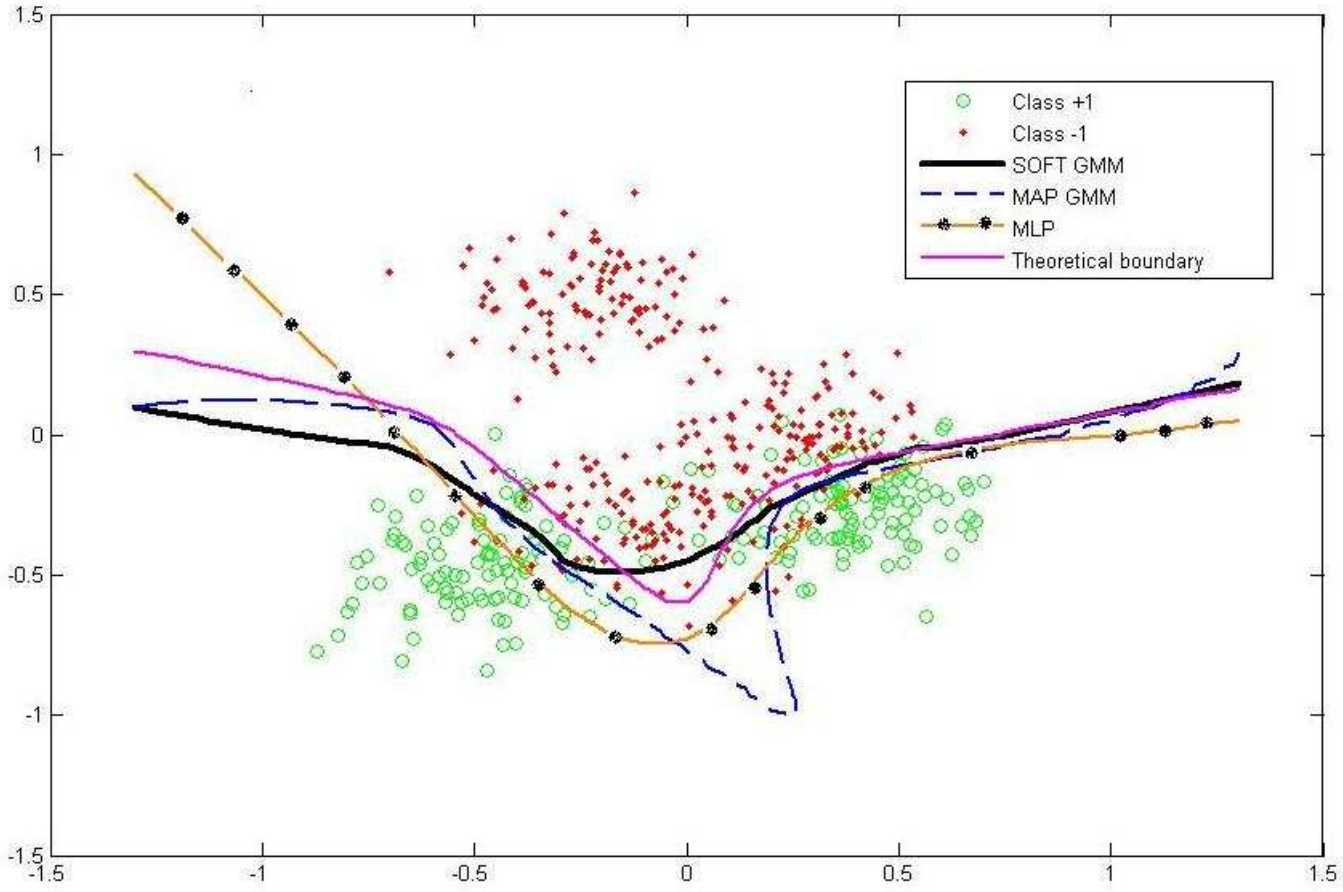

Fig. 3. Boundaries for the three methods mentioned in Table 2 compared with the theoretical boundary for test problem kwo, which is subsampled to 500 samples (C_{+1} : 200/C_{-1} : 300)

gets the omniscient result in two problems (aba and con), while the MAP GMM gets it just once (ion). Additionally, even with different design parameters, the CV and omniscient designs for our method show small performance differences for kwo and bre, and significant differences only for pim, just the problem for which the improvement of our proposal with respect to standard designs is the highest. When applying MAP GMM schemes, small differences also appear for kwo and con, and are important for bre. For MLPs, CV offers the omniscient design in ion, and a significant difference in kwo and pim, even N being the only design parameter to be selected. So, it appears that we have a reduced sensitivity when using our designs.

V. Conclusion

In this work, we have extended the idea of emphasized target smoothing to GMM based decision (it is also possible to extend it to other generative models, such as Parzen windows [34], using Nadaraya-Watson regression [35][36] to deal with soft targets; similar results are obtained). As in the previously studied case of MLP, extensive simulation results allow to conclude that the proposed approach does allow eventual performance improvements with respect to the component standard machines, and it even offers a low sensitivity with respect to the selection of design parameters values. Obviously there is a need for more computational effort to design the proposed classifiers, but their operation is computationally light, and, in the case of being applied to generative models, they have the advantages of allowing a relatively easy interpretation and make it possible to apply principled methods to deal with missing values.

Further research on the extensions of the proposed method will address its use in GP machines, that do not have direct forms for classification purposes because they are based on interpreting targets as Gaussian processes defined on the observations domain.

References

[1] R. A. Fisher, “The use of multiple measurements in taxonomic problems”, Annals of Eugenics, **7**, Pt. II, 179–188, 1936.

[2] S. Y. Kung, J. S. Taur, “Decision-based neural networks with signal/image classification applications”, IEEE Trans. Neural Networks, **6**, 170–181, 1995.

[3] B. A. Telfer, H. H. Szu, “Energy functions for minimizing misclassification error with minimum-complexity networks”, Neural Networks, **7**, 809–818, 1994.
[4] F. Rosenblatt, “The Perceptron: A probabilistic model for information storage and organization in the brain”, Psychological Review, **65**, 386–408, 1958.
[5] P. E. Hart, “The condensed nearest neighbor rule”, IEEE Trans. Information Theory, **14**, 515–516, 1968.
[6] J. Sklansky, L. Michelotti, “Locally trained piecewise linear classifiers”, IEEE Trans. Pattern Anal. Machine Intelligence, **2**, 101–111, 1980.
[7] M. Plutowski, H. White, “Selecting concise training sets from clean data”, IEEE Trans. Neural Networks, **4**, 305–318, 1993.
[8] S. H. Choi, P. Rockett, “The training of neural classifiers with condensed datasets”, IEEE Trans. Sys., Man, and Cybernetics, Pt. B, **32**, 202–206, 2002.
[9] P.W. Munro, “Repeat until bored: A pattern selection strategy”, Adv. in Neural Inf. Proc. Sys. 4 (J. E. Moody et al., eds.), 1001–1008; San Mateo, CA: Morgan Kaufmann, 1992.
[10] C. Cachin, “Pedagogical pattern selection strategies”, Neural Networks, **7**, 171–181, 1994.
[11] Y. Freund, R. E. Schapire, “Experiments with a new boosting algorithm”, Proc. 13th Intl. Conf. Machine Learning, 148–156; Bari (Italy), 1996.
[12] Y. Freund, R. E. Schapire “Game theory, on–line prediction, and boosting”, Proc. 9th Annual Conf. on Comput. Learning Theory, 325–332; Desenzano di Garda (Italy), 1996.
[13] R. E. Schapire, Y. Singer, “Improved boosting algorithms using confidence-rated predictions”, Machine Learning, **37**, 297–336, 1999.
[14] V. Gómez-Verdejo, M. Ortega-Moral, J. Arenas- García, A. R. Figueiras-Vidal, “Boosting by weighting critical and erroneous samples”, Neurocomputing, **69**, 679–685, 2006.
[15] V. Gómez-Verdejo, J. Arenas-García, A. R. Figueiras-Vidal, “A dynamically adjusted mixed emphasis method for building boosting ensembles”, IEEE Trans. Neural Networks, **19**, 3–17, 2008.
[16] B. E. Boser, I. Guyon, V. Vapnik, “A training algorithm for optimal margin classifiers”, Proc. 5th Annual Workshop Comp. Learning Theory (D. Hassler, ed.), 144–152; Pittsburgh, PA: ACM Press, 1992.
[17] C. Cortes, V. Vapnik, “Support Vector networks”, Machine Learning, **20**, 273–297, 1995.
[18] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf “An introduction to kernel–based learning algorithms”, IEEE Trans. Neural Networks, **12**, 181–201, 2001.
[19] L. Franco, S. A. Cannas, “Generalization and selection of examples in feed–forward neural networks”, Neural Computation, **12**, 2405–2426, 2000.
[20] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.
[21] C. E. Rasmussen, C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press, 2006.
[22] R. Reed, S. Oh, and R. J. Marks, II, “Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter”, IEEE Trans. Neural Networks, **6**, 529–538, 1995.
[23] D. Gorse, A. J. Shepperd, J. G. Taylor, “The new ERA in supervised learning”, Neural Networks, **10**, 343–352, 1997.
[24] S. El Jelali, A. Lyhyaoui, A. R. Figueiras-Vidal, “An emphasized target smoothing procedure to improve MLP classifiers performance”, Proc. 16th European Symp. Artificial Neural Networks, 499–504; Bruges (Belgium), 2008.
[25] L. Breiman, “Combining predictors”, *in Combining Artificial Neural Nets.: Ensemble and Modular Multi-net Systems* (A. J. C. Sharkey, ed.), 31–50; London, UK: Springer, 1999.
[26] R. A. Jacobs, M. I. Jordan, “A competitive modular connectionist architecture”, in Advances in Neural Info. Proc. Sys. 5 (D. Touretzky, ed.), 767–773; San Mateo, CA: Morgan Kaufmann, 1991.
[27] M. I. Jordan, R. A. Jacobs, “Hierarchical Mixtures of Experts and the EM algorithm”, Neural Computation, **6**, 181–214, 1994.
[28] L. Xu, M.I. Jordan, G. E. Hinton, “An alternative model for Mixtures of Experts”, in Advances in Neural Information Processing Systems 7, 633–640. MIT Press, 1995.
[29] J. T. Kwok, “Moderating the output of Support Vector classifiers”, IEEE Trans. Neural Networks, **10**, 1018–1031, 1999.
[30] C. L. Blake, C. J. Merty: UCI Repository of Machine Learning Databases: www.ics.uci.edu/∼mlearn
[31] A. Ruiz, P. E. López-de-Teruel, “Nonlinear kernel-based statistical pattern analysis”, IEEE Trans. Neural Networks, **12**, 16–32, 2001.
[32] C. Archambeau, J. A. Lee, M. Verleysen, “On convergence problems of the EM algorithm for finite mixture models”, Proc. 11th European Symposium on Artificial Neural Networks, 99–106; Bruges (Belgium), 2003.
[33] C. Archambeau, F. Vrins, M. Verleysen, “Flexible and robust Bayesian classification by finite mixture models”, Proc. 12th European Symposium on Artificial Neural Networks, 75–80; Bruges (Belgium), 2004.
[34] É. A. Nadaraya, “On estimating regression”, Theory of Probability and Its Applications, **9**, 141–412, 1964.
[35] G. S. Watson, “Smooth regression analysis”, Sankhyā: The Indian Journal of Statistics, Series A, **26**, 259–279, 1964.
[36] E. Parzen, “On estimation of a probability density function and mode”, Annals of Math. Statistics, **33**, 1065–1076, 1962.
[37] F. Pérez–Cruz, “IRWLS Matlab toolbox to solve the SVM for pattern recognition and regression estimation”, 2002. Available: http://www.tsc.uc3m.es/∼fernando/